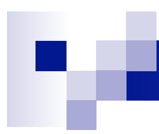
OS Réseaux et Programmation Système - C1

Rabie Ben Atitallah, LAMIH

rabie.benatitallah@univ-valenciennes.fr

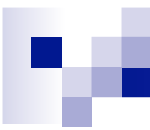
Contributeurs :

Sylvain Lecomte et Marie Thilliez



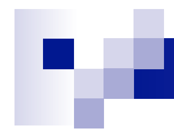
Plan du cours

- Introduction aux systèmes informatiques
  - Présentation générale des SEs
  - Définition
- Un exemple : Unix
  - Quelques commandes de base
- Le Shell
  - Ecriture dans Unix
  - Le Shell de Bourne



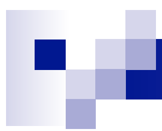
Les systèmes informatiques

- Système informatique
  - Du matériel « hardware »
  - Du logiciel « software »
    - Logiciel de base
      - Outils de dvt et d'exéc de prog
        - Compilateurs
        - Editeurs de texte
        - Chargeurs
      - Système d'exploitation
    - Logiciel d'application



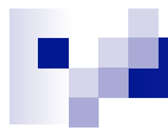
Syst. d'exploitation : définition

- Système d'exploitation (SE) = Operating System (OS) en anglais
- Un SE = Un allocateur et gestionnaire des ressources.
- Besoin de ressources pour exécuter un programme



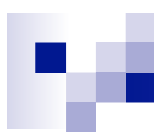
Les ressources

- Ressources matérielles d'un ordinateur :
  - Unité centrale (UC)
  - Mémoires (persistantes ou non)
  - Périphériques d'entrées/sorties

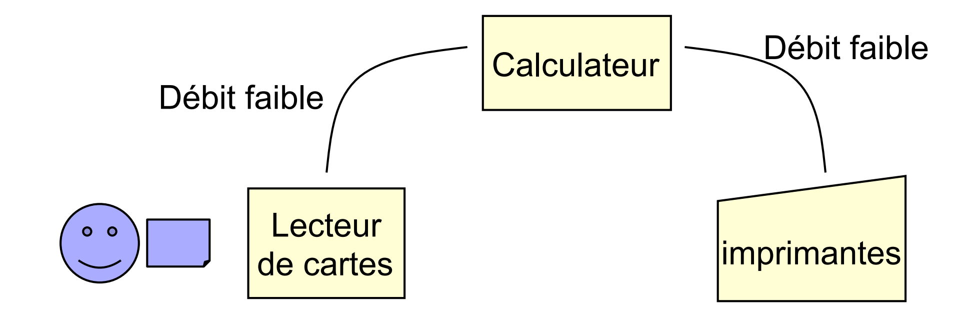


Pourquoi gérer les ressources ?

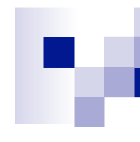
- Les ressources sont limitées
  - Raison économique
  - Raison matérielle
  - Raison de cohérence des données
- Besoin de partager les ressources
  - Les systèmes actuels sont multi-utilisateurs
  - Ils gèrent les ressources pour tous les utilisateurs



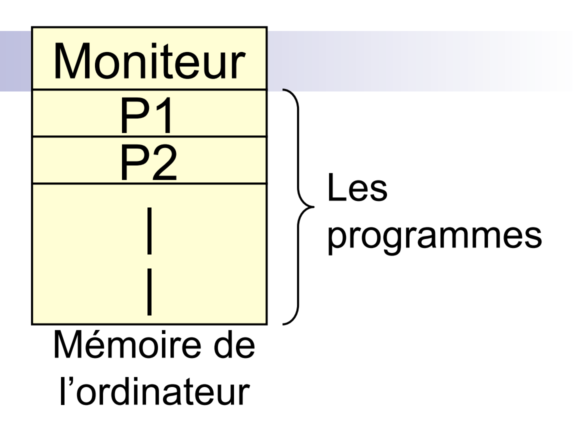
Historique



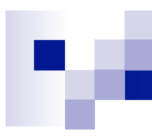
- Premiers systèmes informatiques (45-55)
  - Caractéristiques
    - Du matériel uniquement
    - Pas de système d'exploitation
    - Système mono-usager
  - Problèmes
    - Gestion du système basée sur la réservation de plages horaires
    - Manque de fiabilité du matériel
  - Evolution
    - Périphériques : apparition des dérouleurs de bandes magnétiques
    - Logiciel : Apparitions des premiers outils du logiciel de base (assembleur, chargeurs, compilateurs fortran et cobol)



Historique (suite)

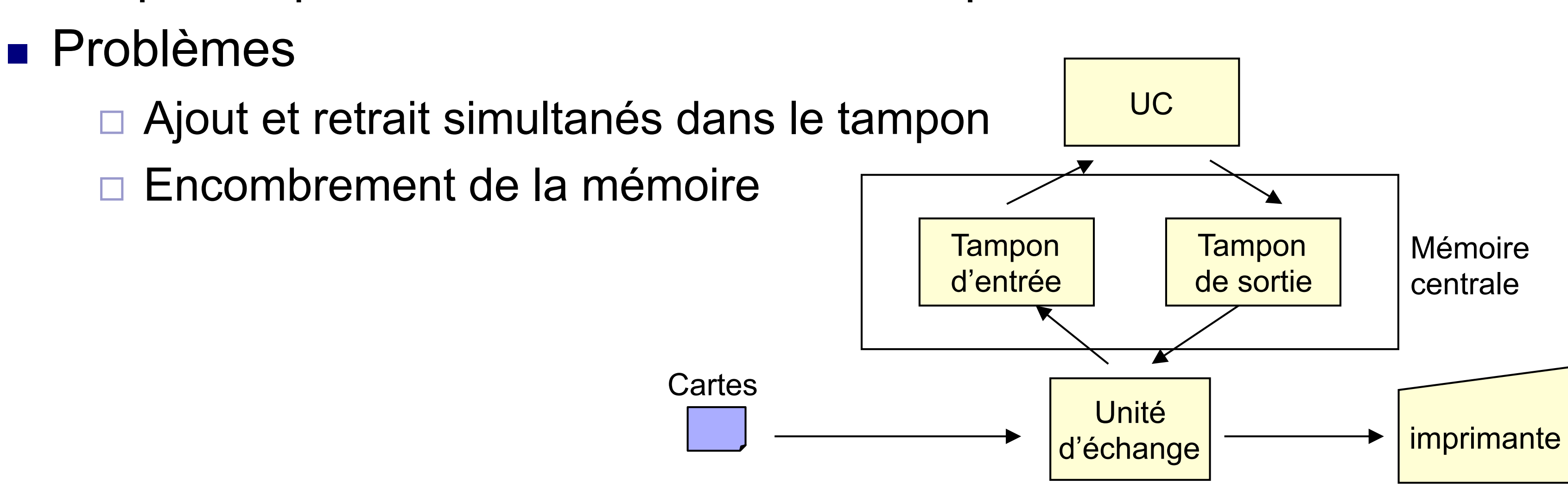


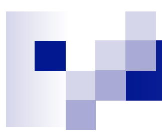
- Les systèmes à moniteurs (55-65) : solution aux pb de réservation et de tps de préparation
  - La technique : Enchaînement automatique des programmes par exécution d'un moniteur
  - Caractéristiques :
    - Système d'exploitation = moniteur
    - Système non-interactif
    - Traitement par lot
    - Système multi-usagers
    - Fonctionnement en mono-programmation : exécution d'un seul programme à la fois
  - Problèmes de protection
    - Comment éviter qu'un programme d'application puisse écrire dans la zone réservée au moniteur ?
    - Comment forcer le programmeur à utiliser les pilotes de périphériques présents dans le moniteur et lui interdire d'agir directement sur les périphériques ?
    - Comment interdire qu'un travail monopolise l'UC ?



Améliorations des systèmes infos

- Problème : la lenteur des périphériques par rapport à l'UC
  - Les E/S tamponnées : utilisation d'unités d'échange (UE) capables de fonctionner simultanément avec l'UC.
    - Principe : les cartes sont lues par l'UE et stockées dans des tampons (buffers) d'entrée. L'UC lit les données dans le tampon et produit le résultat dans le tampon de sortie.
    - Problèmes
      - Ajout et retrait simultanés dans le tampon
      - Encombrement de la mémoire



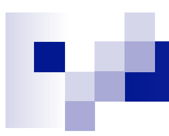


Améliorations (suite)

- Les E/S spoolées
  - Principe : Tampons en mémoire secondaire
  - Problème : l'UC est contrainte d'attendre la terminaison des opérations d'E/S
  - Solution : la multi-programmation

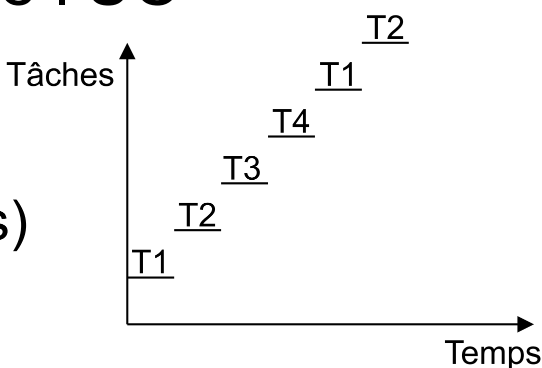
    Quand l'UC se trouve en attente d'E/S, elle suspend le programme en cours et reprend l'exécution d'un autre programme.

    Donc plusieurs programmes résident simultanément en mémoire.

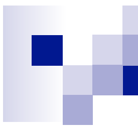


Les systèmes à temps partagé

- Systèmes interactifs multi-usagers fonctionnant en multiprogrammation avec partage de l'UC (« time sharing »). Ex : UNIX
- Principe : considérer que l'UC est une ressource et l'allouer durant un temps limité : partage de l'UC
- Problèmes :
  - Gestion des périphériques
  - Gestion des mémoires (centrale et secondaires)
  - Gestion des erreurs



Nécessité d'un ensemble de programmes (SE) pour résoudre ces problèmes !!!

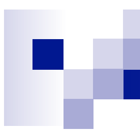


Un SE = une machine virtuelle

- SE = Réalisation d'une machine virtuelle au dessus de la machine matérielle

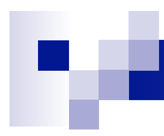
  permettant au programmeur de s'abstraire des détails de mise en œuvre du matériel

- Notion de machine virtuelle
  - Traduction : Analyser chaque instruction d'un programme P_i écrit en L_i et la remplacer par la séquence d'instructions équivalentes dans le langage L_{i-1}.
  - Interprétation : Ecrire dans le langage L_i, un programme I capable d'analyser, une à une, chaque instruction d'un programme L_{i+1} et exécuter immédiatement la séquence d'instructions L_i équivalentes. I est appelé interpréteur.
  - Seule contrainte : « respect de la hiérarchie ». Un programme s'exécutant sur la machine M_i ne peut être traduit ou interprété en instructions d'un langage L, tel que L soit associé à une machine M_j avec $i<j$.



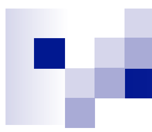
Notion de machine virtuelle

Machine virtuelle M_n Langage L_n	Les programmes écrits en L_n sont traduits en L_{n-1} ou interprétés par un programme s'exécutant sur M_{n-1}.
↓ Traduction ou interprétation	
Machine virtuelle M_1 Langage L_1	Les programmes écrits en L_1 sont traduits en L_0 ou interprétés par un programme s'exécutant sur M_0.
↓ Traduction ou interprétation	
Machine virtuelle M_0 Langage L_0	Les programmes écrits en L_0 sont directement exécutés sur M_0.

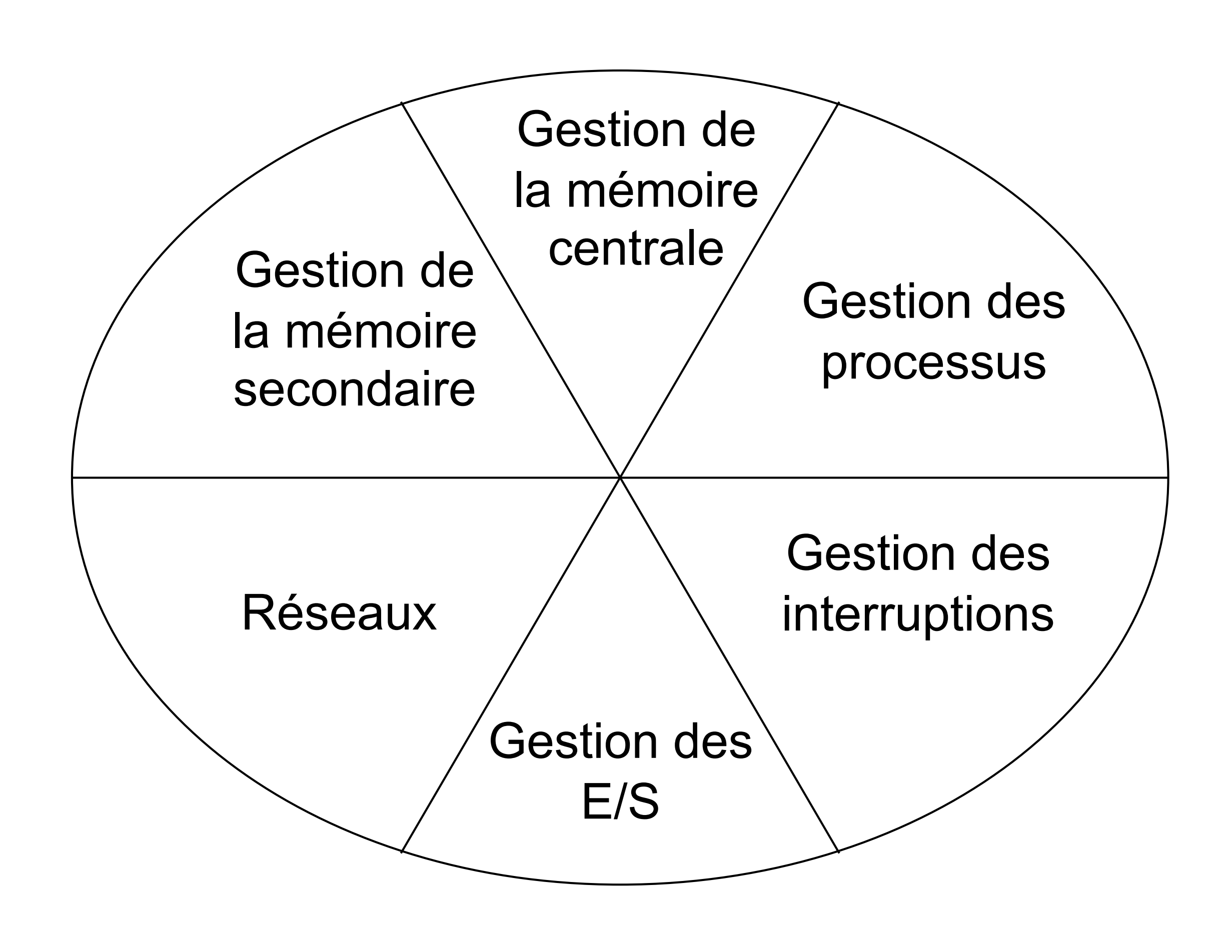


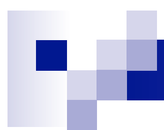
Les systèmes multicouches

Couche	Passage
Couche Langages évolués	
↓	Traduction (compilateur)
Couche Assemblage	
↓	Traduction (assembleur)
Couche SE	
↓	Interprétation partielle (SE)
Couche Machine	
↓	Interprétation (microprogramme)
Couche microprogrammée	
↓	Traduction
Couche physique	Exécution directe des micro-instructions par le matériel



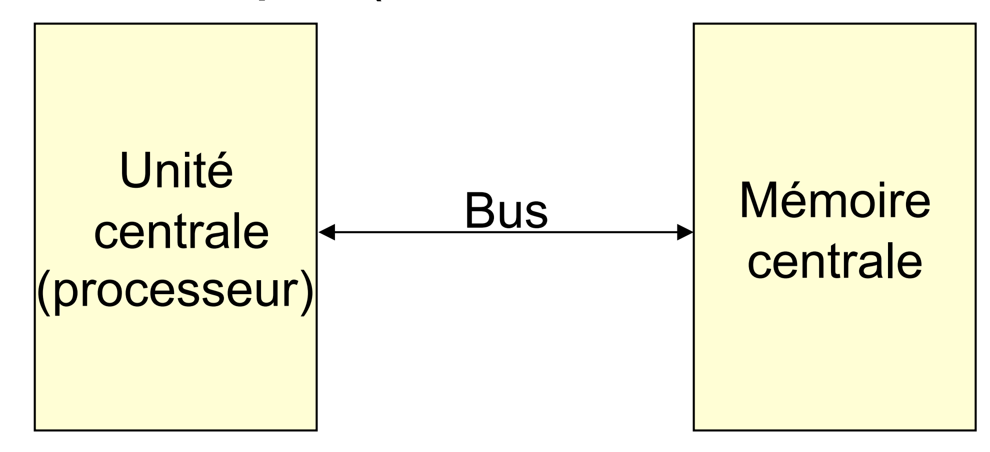
Structure du SE



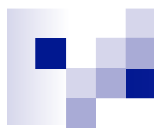


Architecture de la couche physique

- Modèle classique (modèle Von Neumann, 1945)

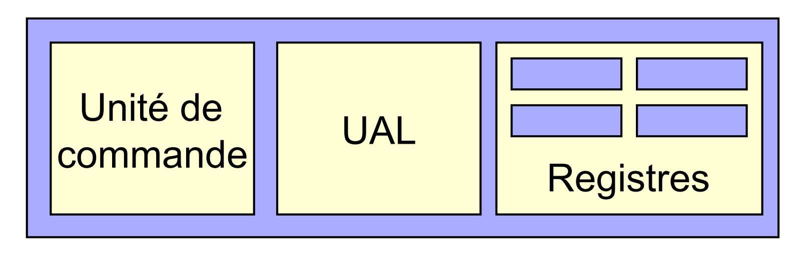


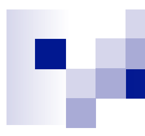
- Le rôle de l'unité centrale (UC) est d'exécuter les programmes stockés dans la mémoire principale
- La mémoire contient les programmes et les données :
  - Les mémoires volatiles (RAM Random Access Memory)
  - Les mémoires mortes (ROM Read Only Memory)



Structure interne d’un processeur

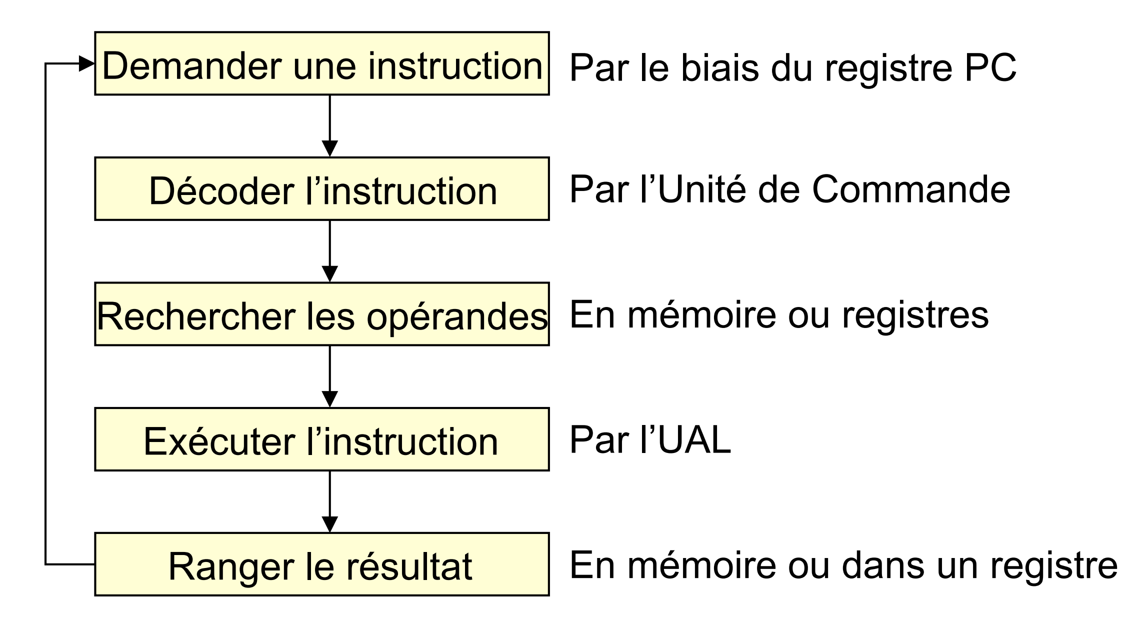
- Exemple de processeurs
  - Famille Intel : celeron, pentiums
  - Famille Motorola : sparcs
- Jeu d’instructions d’un processeur : lang. machine
  - Ensemble d’instructions que peut exécuter le processeur
  - Langage de programmation de plus bas niveau
- Types d’instructions
  - Addition de 2 nombres
  - Tests (très élémentaires)
  - Accès en mémoire (écrire et lire un nombre en mémoire)
- Structure interne d’un processeur
  - Unité de commande : charge une instruction et la décode
  - Unité Arithmétique et Logique (UAL) : exécute les opérations
  - Registres : mémoires à accès très rapide qui permettent de stocker des résultats temporaires ou des informations de contrôle

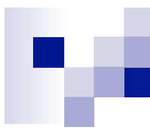




Exécution d'un programme

- Pour exécuter un programme, l'UC dispose :
  - D'un registre PC (compteur ordinal ou de programme) : il indique l'endroit en mémoire principale de la prochaine instruction à exécuter
  - D'un registre d'instructions RI qui contient le code de l'instruction à exécuter
  - D'une UAL (ou ALU en anglais)
  - De diverses registres
- Exécution d'un programme

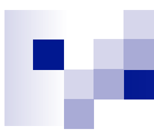




Systèmes informatiques actuels

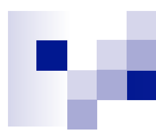
- Systèmes des ordinateurs personnels (PC, PDA etc.)
  - Mono-usager
- Systèmes à temps partagé
- Systèmes de commandes de procédés
  - Périphériques + capteurs
  - Contrainte de temps réel : temps de réponse borné (très court) garanti qq soit l'activité du système.
- Systèmes à transaction
  - Gèrent des bases de données de grande taille
  - Mise à jour de la base par des transactions
- Systèmes multiprocesseurs
  - But : Hautes performances
  - Le système gère l'allocation de plusieurs UCs
- Systèmes répartis
  - Facilitent l'exécution répartie
  - Buts : partage des ressources, accélération du calcul, fiabilité et communication
- Systèmes réseaux
  - Permet aux utilisateurs des stations de travail reliés par un réseau de partager des ressources communes, par exemple un système de fichiers
  - Ex : NFS (Network File System)

Unix



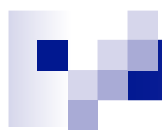
Historique

- Ken Thompson, Dennis Ritchie (Bell Labs, 1969)
- Système d'exploitation portable
  - Écriture d'un système portable
  - Donc définition du langage C : très simple, très portable mais proche de l'assembleur pour être efficace.
  - Noyau écrit à 90% en C
- Adoption par les universitaires
  - Sources disponibles
  - Support pour les cours systèmes
- Succès progressif dans l'industrie
  - Robuste, ouvert, portable



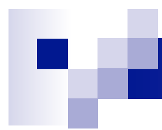
Aujourd'hui

- Puissance
  - PC au super-calculateurs (Cray)
- Portabilité
  - Très nombreuses plateformes (Sparc, Alpha, PowerPC, …)
- Compatibilités
  - Normes : X/Open et IEEE POSIX
  - API : SYSV et BSD (scission en 1979)
  - Objet : ELF, COFF
- Versions
  - Payantes : Solaris, AIX, DECUnix, SCO, HPUX …
  - Libres de droit : Linux, FreeBSD, …

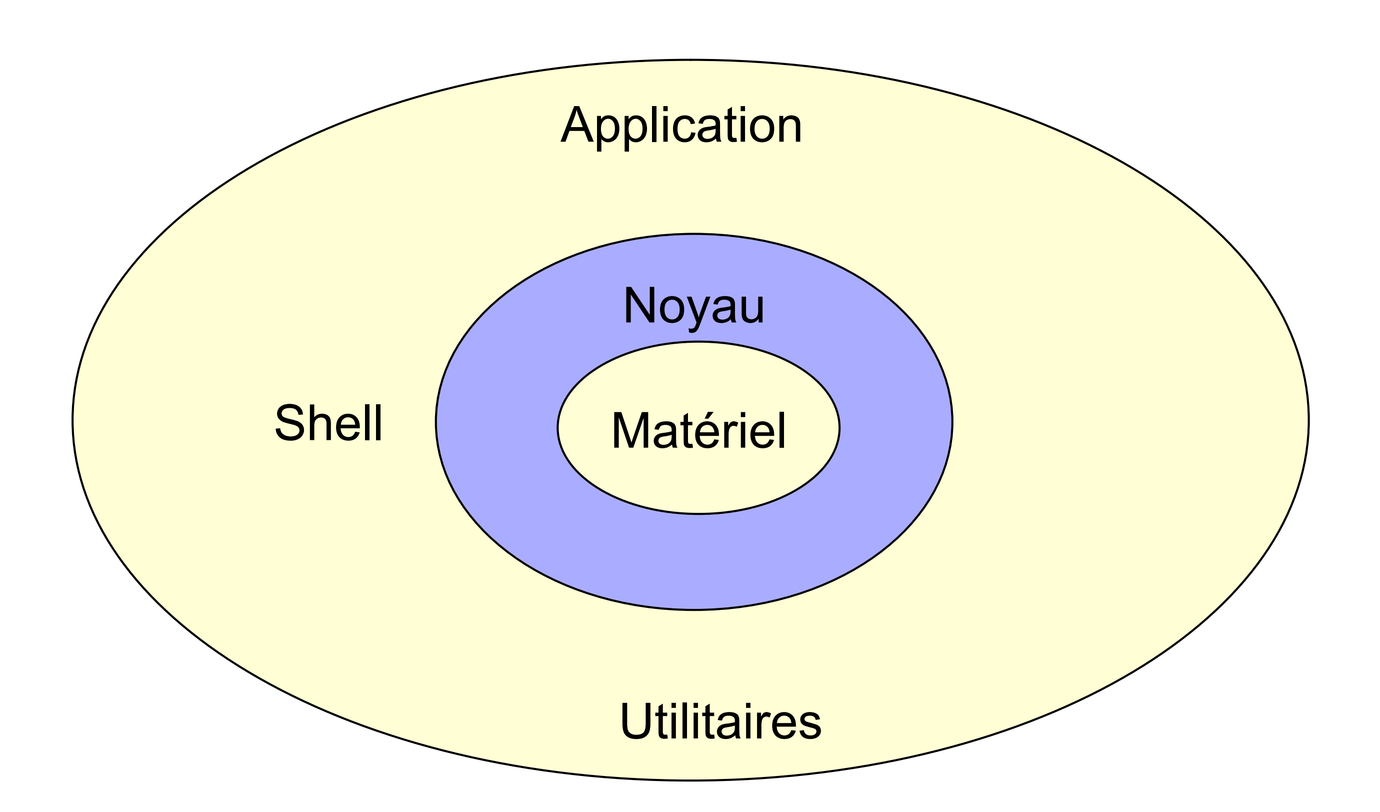


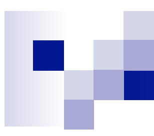
Caractéristiques d’Unix

- Portabilité
- Multi-utilisateurs
- Multi-tâches
- Interactif
- Système de fichiers hiérarchisé
- Mécanisme de protection
- Vision simplifiée des E/S par l’utilisateur
- Choix d’un langage de commandes : les shells



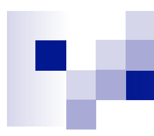
Structure du système Unix (i)





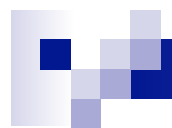
Structure (ii)

- Le noyau gère les tâches de base du système
  - Initialisation du système
  - Gestion des ressources
  - Gestion des processus
  - Gestion des fichiers
  - Gestion des E/S
- L'utilisateur (les applications) communique avec le noyau par l'intermédiaire d'un Shell.
- Les shells sont aussi des langages de commandes et de programmation
  - Shells les plus connus : BOURNE SHELL, C-SHELL, KORN-SHELL, TC-SHELL
- Les utilitaires = outils d'interfaçage avec le système, de programmation et de communication.



Les utilisateurs

- UID (User Identifier)
  - Unique dans le système
  - Identifie l'utilisateur
- Username
  - Utilisé au moment du login
  - Donne un environnement particulier à un utilisateur (UID)
- Root = le super utilisateur
  - UID = 0, Username = root
  - Administrateur du système
  - Possède des droits particuliers pour administrer les systèmes

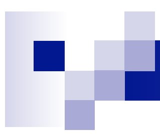


Quelques commandes simples

- date : pour obtenir la date
- cal : pour obtenir le calendrier
- who : qui est connecté ?
- logname : qui suis-je ?
- pwd : où suis-je ?
- tty : nom du terminal
- passwd : Changer le mot de passe

Les Langages de Commande (Shell)

- Lancement des travaux
  - commandes, programmes utilisateurs, …
- 3 modes de fonctionnement
  - Interactif
    - boucle interactive avec l'utilisateur (prompt $)
  - Batch
    - enchaînement de commandes (structure de contrôle, variables, code de retour ...)
  - En arrière plan (&)
    - plusieurs commandes en parallèle
- Ex: `sh`, `csh`, `ksh`, `tcsh`, `bash`, etc



Les commandes

- Des programmes standards du système (également appelés Utilitaires)
  - ls, ps, kill, find, rm, rmdir, mail, who, talk …
  - sh, csh, ...
- Vos propres programmes
  - monls, dir, ex2.exe
  - mysh
- Les programmes des autres utilisateur
  - lscommun
- La plupart des commandes proposent des options

Exemples…

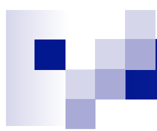
- pwd

↳ `/root/src/system/Shells`

- ls -l

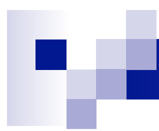
↳ total 12

```
-rwx------   1 root     root           60 Nov 18  2001 add*
-rwx------   1 root     root          122 Nov 18  2001 boucle*
-rwx------   1 root     root           91 Nov 18  2001 f*
-rw-r--r--   1 root     root            0 Nov 11 16:51 ls
-rwxr-xr-x   1 root     root          300 Nov 18  2001 occ*
drwxr-xr-x   2 root     root         1024 Nov 18  2001 old/
-rwx------   1 root     root           81 Nov 18  2001 rof*
-rwx------   1 root     root          293 Nov 18  2001 taille*
-rwx------   1 root     root          169 Nov 17  2001 taille1*
drwxr-xr-x   2 root     root         1024 Nov 18  2001 test/
-rwx------   1 root     root          562 Nov 18  2001 tmax*
-rwx------   1 root     root          194 Nov 18  2001 tmax2*
-rw-r--r--   1 root     root           24 Oct  5 16:32 tmp
```



Les Processus

- Contexte d 'exécution d 'une tâche
  - pas de partage des ressources avec les autres processus à part sur les fichiers
  - le code de la tâche est dans un fichier
- 2 types de Processus
  - Processus utilisateur
    - tâche lancée par un utilisateur
  - Processus Système
    - tache lancée par `root` pour les besoins du système
      - lpd : spool d 'impression,
      - nfsd : partage de fichier,
      - rlogind, telnetd : login distant, …



Propriétés des Processus

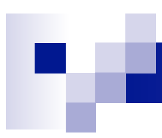
- Identifiant
  - Identifiant PID (Process ID)
    - Unique dans le système (du *boot* au *shutdown*)
  - Identifiant du parent (PPID (Parent Process ID)

- Propriétaire
  - UID de l 'utilisateur qui a lancé le processus
  - changement : appel système
    - seulement pour `root`

Création d'un processus (i)

- Depuis un processus parent (mécanisme général)
  - par clonage
    - appel système fork()
  - puis par mutation du code
    - appel système execv()
    - code du programme dans un fichier
- Une exception

  processus 1 : le premier qui engendre tous les autres
    - PID=1, PPID=1



Création d’un processus (ii)

- Depuis un processus *shell*
  - en premier plan (*foreground*)
    - le shell attend la terminaison du processus

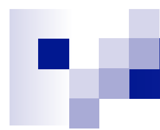
```
> grep donsez /etc/passwd > tmp
> more tmp
```

  - & lancement de processus en mode détaché (ou arrière plan ou *background*)
    - permettre à un utilisateur d ’avoir plusieurs taches actives simultanément (multi-taches)

```
> emacs & ;
```

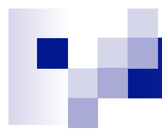
  - | pipelining de processus

```
> grep donsez /etc/passwd | more
> find . -name "*.c" | grep src | wc -l
```



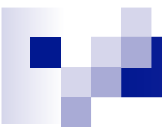
La terminaison d’un Processus

- La terminaison
  - appel système exit()
    - retourne le statut à l ’appelant qui est en attente (wait())
  - commande kill et appel système kill()
    - la terminaison est le comportement par défaut

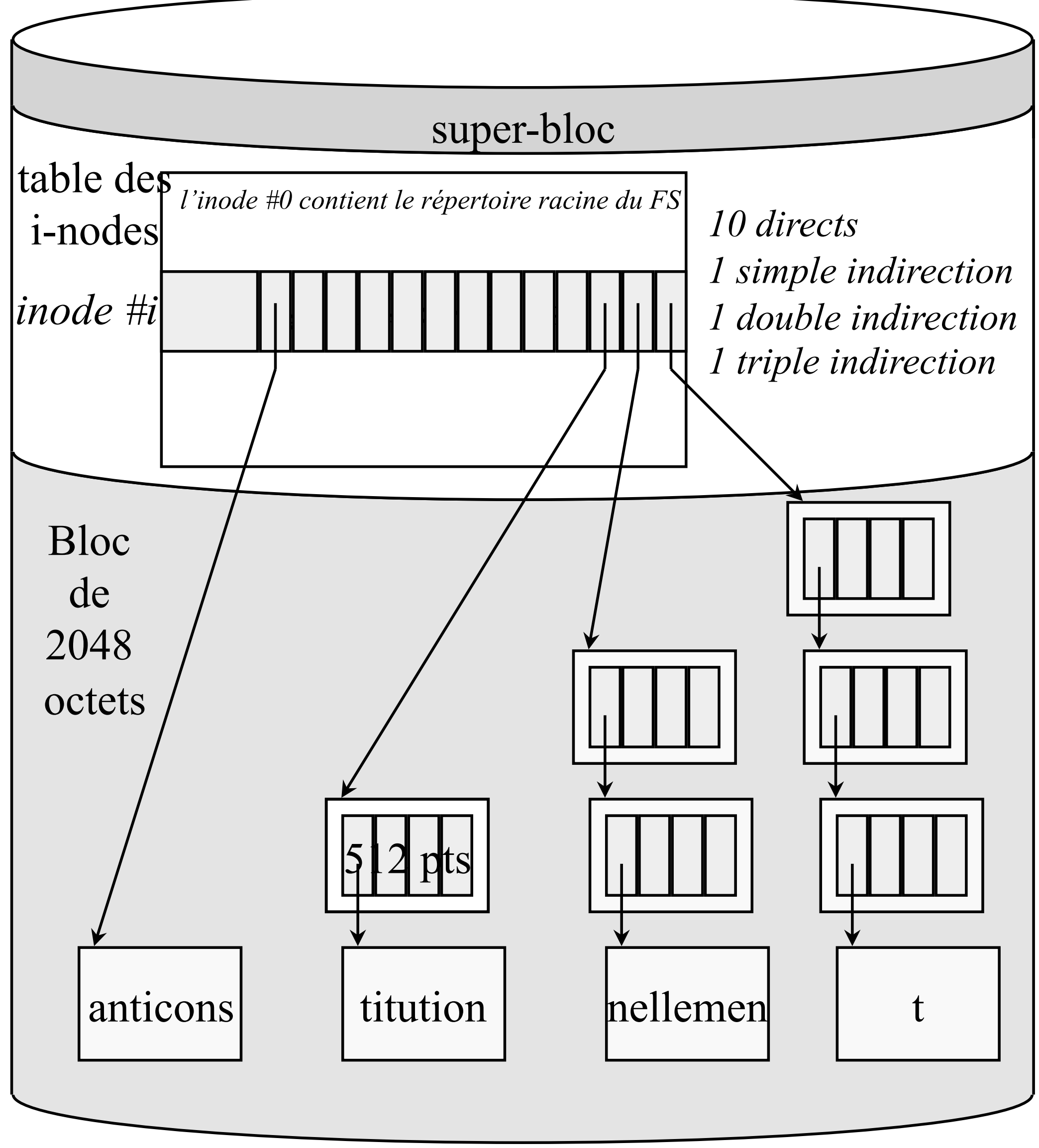


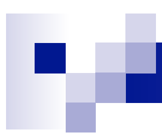
Le Système de Fichiers (i)

- Les « inodes »
  - un inode = contenu d'un fichier ou d'un répertoire
  - un lien = une entrée d'un répertoire désignant un fichier
- Répertoire
  - ensemble de liens (= [nom, numéro d'un inode])
  - ls -i . liste les inodes référencés depuis le répertoire .
- Table des inodes
  - UID, droit, dates, …
  - 10 références disque vers les 10 premiers blocs
  - 3 références disque vers des blocs d 'indirection
    - jusqu 'à 3 niveaux d'indirection



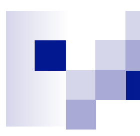
Le Systèmes de Fichiers (ii)





Les fichiers spéciaux

- Les organes d'entrée-Sortie
  - représentation des périphériques en mode brute (*raw*)
  - tous listés dans le répertoire `/dev/`
  - 2 types
    - b(loc)
      - Disque Dur, CD-ROM, Cassette, ...
    - c(aractère)
      - Imprimante, clavier, souris, écran, …
- Les liens symboliques
  - redirection vers une entrée dans un répertoire



Principales commandes sur les fichiers

- Opérations de base sur les fichiers
  - Visualisation du contenu d’un fichier
    - cat <nom_fichier>
    - more <nom_fichier>
  - Renommage et déplacement d’un fichier
    - mv <source> <destination>
      - mv essai.c tp1.c (renommage)
      - mv tp TPSE (déplacement)
      - mv tp TPSE/tp2 (déplacement et renommage)
  - Copie d’un fichier
    - cp <source> <destination>
  - Création de liens sur un fichier
    - ln <ancien> <nouveau>
  - Suppression d’un fichier
    - rm <nom_fichier>
    - Ex : rm *.o : supprime tous les fichiers d’extension « .o

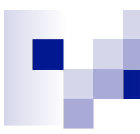
      rm a* : supprime tous les fichiers dont le nom commence par « a »

      rm * : supprime tous les fichiers

      rm *.? : supprime tous les fichiers ayant une extension d’une lettre

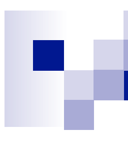
    * : une chaîne quelconque de caractères

    ? : un caractère quelconque



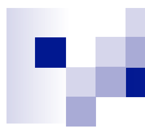
Principales commandes sur les fichiers

- Opérations de base sur les répertoires
  - Création/suppression d’un répertoire
    - mkdir <nom_répertoire>
    - rmdir <nom_répertoire> : supprime un répertoire vide.
    - rm -R <nom_répertoire> : supprime tout le répertoire
  - Montage d’un système de fichiers
    - Possibilité d’ajouter un système de fichiers extérieurs en l’insérant dans la hiérarchie.
    - Les commandes *mount* et *unmount* permettent de monter et de démonter un système de fichiers.
    - L’ajout d’une mémoire de masse est transparent pour l’utilisateur.



Principales commandes sur les fichiers

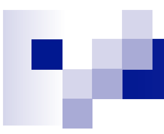
- Affichage du contenu d'un répertoire :
  - ls [options] <nom_répertoire>
  - L'option –l permet d'obtenir l'ensemble des informations relatives à chaque fichier du répertoire :
    - Type de fichier : « - » (fichier ordinaire), « d » (répertoire) ou « b », « c » (fichiers spéciaux)
    - Droits d'accès
    - Nom du propriétaire
    - Taille
    - Nom
    - Date de création
    - Etc…
  - L'option –R permet d'afficher récursivement le contenu d'un répertoire.
    - Ex : ls / (afficher les répertoire à la racine
      - Dev bin usr users etc unix …
    - ls –R /
      - /dev :
liste des fichiers dans /dev
      - /bin :
liste des fichiers dans /bin
      - /users :
/deustiosi :
/deustiosi1 :
….
/deustiosi2 :
…



Autres commandes utiles

touch : crée un fichier
wc : donne le nombre de caractères (-c), de mots (-w) ou de lignes (-l)
sort : permet de trier par ordre alphabétique les lignes d'un fichier
grep : recherche d'un motif dans un fichier
Exemple : grep printf essai.c
grep –l print f *.c (affiche la liste des fichiers contenant « printf »)
head : affiche les première lignes
tail : affiche les dernières lignes
diff : permet de comparer 2 fichiers
find : permet de rechercher un fichier
lpr : imprime un fichier
lpq : affiche les fichiers en attente d'impression
lprm : détruit des fichiers en attente d'impression

man (très utile !!) : donne le manuel d'utilisation d'une commande

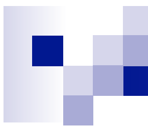


Droits d’accès aux fichiers (ii)

- Droit d ’accès
  - 3 catégories d ’utilisateur

u	u	u	g	g	g	o	o	o
r	w	x	r	w	x	r	w	x

    - u(ser) *le propriétaire du inode*
    - g(roup) *les utilisateurs appartenant au groupe du inode*
    - o(ther) *les autres utilisateurs de la machine*
  - 3 types d’opérations sur les fichiers
    - r(ead) *lire*
    - w(rite) *écrire, ajouter, supprimer*
    - x(eXecute) *exécuter le programme contenu dans le fichier*
  - 3 types d’opérations sur les répertoires
    - r(ead) *lister*
    - w(rite) *ajouter un nouveau fichier*
    - x(eXecute) *parcourir*
- 2 syntaxes : symbolique et octal



Protection des fichiers : mode symbolique

chmod <qui> <permission> <opération> <fichier>

<qui> valant **u**, **g**, **o** ou **a** (pour All/Tous)

<permission> valant **+** pr autoriser, **-** pr interdire

<operation> valant **r**, **w** ou **x**

Exemples :

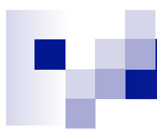
Exercice :

1- spécifier que les membres du groupe peuvent écrire dans le fichier montp.c

chmod g+w montp.c (les membres du groupe peuvent écrire dans le fichier montp.c)

2- Protection du fichier en lecture, en écriture et en exécution pour tout le monde (hors mis le propriétaire)

chmod og-rwx montp.c (protection en lecture, en écriture et en exécution)



Protection des fichiers : mode octal

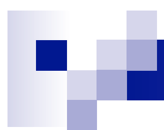
chmod <permission> <fichier>

permission : UGO (User, Group, Others : chiffre octal codant les bits r w x)

u	u	u	g	g	g	o	o	o
r	w	x	r	w	x	r	w	x

Exemples :

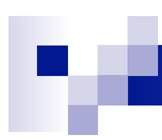
chmod 740 montp.c (rend le fichier accessible en lecture au groupe et inaccessible aux autres)



Droits d'accès aux fichiers

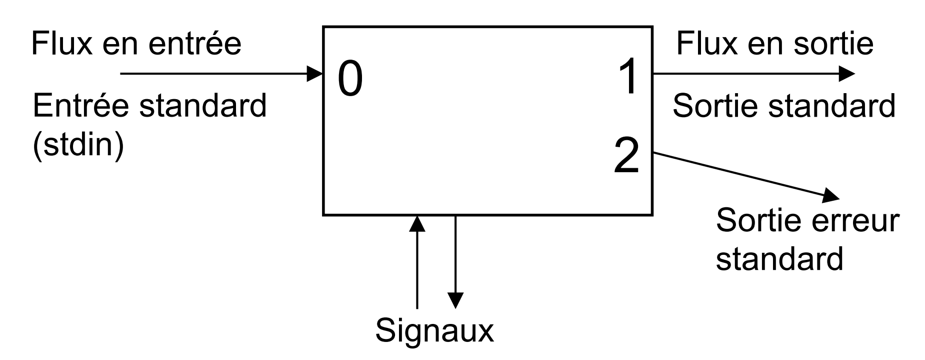
- Propriété d'un Fichier
  - UID du créateur (qui en devient le propriétaire)
  - Chown : changement de propriétaire (*owner*)

- Groupe d 'un fichier
  - GID du créateur
  - Chgrp : changement de groupe



Les fichiers standards et leur redirection

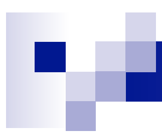
On peut assimiler un processus à un objet abstrait qui dispose de 3 flux d’informations :



Redirection de l’entrée standard : commande < nom_fichier
(l’entrée standard est redirigée sur le fichier de référence nom_fichier)

Redirection de la sortie standard : commande > nom_fichier ou commande >> nom_fichier (redirection sans écrasement)

Redirection de la sortie en erreur standard :
commande 2> nom_fichier (en Bourne Shell)
commande >& nom_fichier (en C-schell)



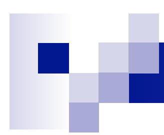
Les fichiers et les processus

- Chaque processus dialogue avec l ’extérieur via des descripteurs de fichiers ouverts
  - Fichier

    ```
    > grep donsez /etc/passwd > tmp
    > more tmp
    ```

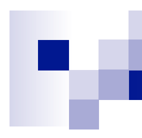
  - Pipe
    - fichier créé temporairement sans existence sur les disques

    ```
    > grep donsez /etc/passwd | more
    ```



Conclusion

- Robuste, Ouvert, Portable
- Offre large gratuite et payante
- Administration système
- Système Réseau
  - TCP/IP, UDP/IP
- Systèmes de Fenêtrage X11 (MIT)
  - Motif, Openwin
- Son principal rival
  - MicroSoft Windows NT



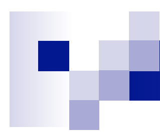
Bibliographie



- Jean-Marie Rifflet, **La programmation sous Unix**, 3ème édition, 630 pages, EdiScience.
  - utilisateur et programmation système

- Maurice Bach, **Conception du système Unix**
  - détail de l'implantation du noyau

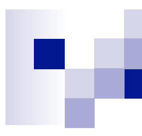
Les SHELL



Introduction

- Shell
  - interface entre l'utilisateur et UNIX

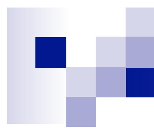
- Les différents types de shells
  - Bourne shell (prompt par défaut $)
  - C shell (prompt par défaut %)
  - Korne shell (prompt par défaut $)



Introduction (suite)

- Shell est un programme (/bin/sh) qui interprète et exécute les commandes :
  - Création des processus
  - Attente des fins d'exécutions
  - Redirection des E/S : stdin, stdout et stderr
- Shell est aussi un langage de commandes
  - Avec interprétation de phrases conditionnelles, composées ou itératives (if … then …else, for …, while …)
- Shell peut lire ses commandes dans un fichier appelé procédure (Script-shell)
- Shell gère un certain nombre d'indicateurs et de variables utilisables par le programmeur
- Shell autorise le passage de paramètres

Le Bourne shell



Caractères spéciaux

* : une chaîne quelconque

? : n'importe quel caractère

< > | & : redirection

\n : permet d'annuler l'effet d'un caractère spécial n

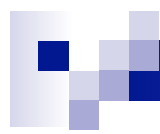
[] : un caractère spécial de l'intervalle

ex: [0 1 2 3 4]

ls * [0-9]

Les variables d'environnement

- Les plus connues :
  - HOME : le répertoire de login
  - LOGNAME : le nom de login
  - PS1 : le message d'appel principal
  - PATH : la liste des répertoires où le shell recherche les commandes à exécuter
  - TERM : indique l'émulation du terminal
- La valeur s'obtient à l'aide de $:
  - exemple :
    - echo $PATH
    - echo $LOGNAME
- Affectation par le signe =
  - exemple : PS1 = nom message
    PS1 = $LOGNAME



Manipulation des variables

- Lecture d’une valeur au clavier

```
read var
```

- Affichage d’une variable

```
echo $var
```

- Un petit exemple…

```
read a b c
Ceci est un exemple
```

`echo $a`

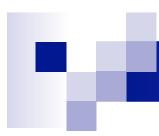
Ceci

`echo $b`

est

`echo $c`

un exemple



Manipulation des variables (2)

- Opérations arithmétiques

  a = 100

  a ~~= $a +~~ 1

  a = `expr $a + 1`

  echo $a

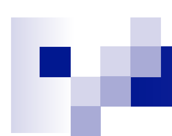
  Appels de commandes UNIX

  a = `pwd`

  echo $a

  /usr/local/toto

- Attention
  - Les `` sont utilisés pour interpréter des commandes et non pour déclarer des chaînes de caractères

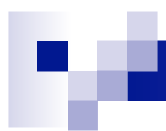


Shell-script ou procédure de commandes

- Un shell script est un fichier texte contenant une liste de commande
  - exemple :

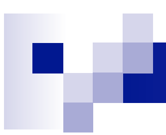
```
cat MonScript
echo 'Bonjour'
echo $LOGNAME
pwd
...
```

- NB: le nom d'une procédure est celui du fichier qui la contient



Exécution d'un shell

- Pour exécuter ce shell, 2 solutions :
  - $ `sh MonScript` (interpréteur de commande shell)
  - $ `chmod 711 MonScript`
    $ ./`MonScript`
- Pour être exécuter un programme shell, il faut avoir des droits en exécution (--x--x--x)

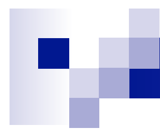


Paramètres d’un shell

- Le passage d’arguments :
  - Se fait via les variables $0 à $9

```
commande    arg1 arg2 arg3 arg4
   $0        $1   $2   $3   $4
```

  - $# représente le nombre d’arguments de la commande
  - $* représente l’ensemble des arguments déjà interprétés
  - $? contient le code de retour de la commande (valeur ≠ 0 si erreur)



Un petit exemple…

$cat exemple

echo $0

echo $#

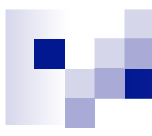
cp $1 $2

exemple *.c repertoire

exemple

2

#tous les fichiers avec l’extension .c sont copiés dans répertoire



Procédure shell : exemple 2

```
$ cat com_sh
echo Nom de la commande: $0
echo La commande a $# arguments
echo Liste des arguments : $*
echo $1 $3
```

← Affichage de la procédure (for `$ cat com_sh`)

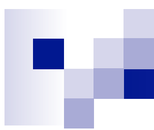
} Procédure (for the four `echo` lines)

```
$ com_sh a b c d e f
Nom de la commande: com_sh
La commande a 6 arguments
Liste des arguments: a b c d e f
a c
```

← Lancement de la procédure (for `$ com_sh a b c d e f`)

} Résultat de l'exécution de la procédure (for the four output lines)

Les structures de contrôle en Shell

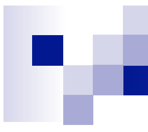


La commande test

- Manipulation des fichiers

```
test <option> <fichier>
```

- Les options :
  - -f : code retour = 0 si fichier existe et est de type ordinaire
  - -d : code retour = 0 si fichier existe et est de type répertoire
  - -r : code retour = 0 si fichier existe et est accessible en lecture
  - -w : code retour = 0 si fichier existe et est accessible en écriture
  - -x : code retour = 0 si fichier existe et est exécutable
  - -s : code retour = 0 si fichier existe et n’est pas vide

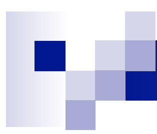


Exemple…

- En ligne de commande

```
test -f monFichier
echo $?
0
```

- Utilisable également dans un shell, notamment dans une alternative.

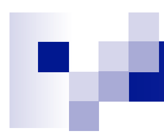


La commande test (2)

- Numériques et chaînes de caractères

```
        test <opd1> <operateur> <opd2>
ou      [<opd1> <operateur> <opd2>]
```

- Numériques
  - -eq : égalité
  - -ne : différence
  - -gt : +grd que
  - -ge : +grd ou égal
  - -lt : +petit que
  - -gt : +petit ou égal
- Chaînes de caractères
  - = : égal à
  - != : différent de



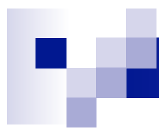
Alternative

- Instruction if

  if <liste de commandes>
  then <liste de commandes>
  else <liste de commandes>
  fi

- Exemple

```
if test -d $1
 then echo le fichier est un repertoire
 else echo le fichier n est pas un
repertoire
fi
```



Case

- instruction case

  case mot in

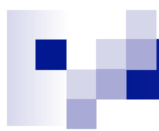
  <motif1> <liste de commandes> ;;

  <motif2> <liste de commandes> ;;

  …

  * <liste de commandes à exécuter par défaut> ;;

  esac

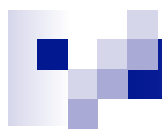


Répétition

- Pour chacune des valeurs d'un ensemble
  - Instruction for

```
for <variable> in <chaine 1> ... <chaine n>
      do
            <liste de commandes>
      done
```

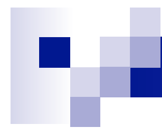
  - $variable prend successivement les valeurs de `chaine 1` à `chaine n`



Répétition (2)

- Tant que le code de retour de la dernière cde est nul, exécuter liste de cde
  - Instruction while

```
while <liste de commandes>
  do
      <liste de commandes>
  done
```



La commande shift

- Décalage des paramètres

```
while test $# -ne 0
   do
    echo $1
    shift
 done
```